MORRISON & FOERSTER LLP
MICHAEL A. JACOBS (Bar No. 111664)
mjacobs@mofo.com
MARC DAVID PETERS (Bar No. 211725)
mdpeters@mofo.com
DANIEL P. MUINO (Bar No. 209624)
dmuino@mofo.com
755 Page Mill Road, Palo Alto, CA 94304-1018
Telephone: (650) 813-5600 / Facsimile: (650) 494-0792

BOIES, SCHILLER & FLEXNER LLP
DAVID BOIES (Admitted *Pro Hac Vice*)
dboies@bsfllp.com
333 Main Street, Armonk, NY 10504
Telephone: (914) 749-8200 / Facsimile: (914) 749-8300
STEVEN C. HOLTZMAN (Bar No. 144177)
sholtzman@bsfllp.com
1999 Harrison St., Suite 900, Oakland, CA 94612
Telephone: (510) 874-1000 / Facsimile: (510) 874-1460

ORACLE CORPORATION
DORIAN DALEY (Bar No. 129049)
dorian.daley@oracle.com
DEBORAH K. MILLER (Bar No. 95527)
deborah.miller@oracle.com
MATTHEW M. SARBORARIA (Bar No. 211600)
matthew.sarboraria@oracle.com
500 Oracle Parkway, Redwood City, CA 94065
Telephone: (650) 506-5200 / Facsimile: (650) 506-7114

Attorneys for Plaintiff
ORACLE AMERICA, INC.

UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA

SAN FRANCISCO DIVISION

ORACLE AMERICA, INC. Plaintiff, v. GOOGLE INC. Defendant.	Case No. CV 10-03561 WHA **ORACLE'S RESPONSIVE BRIEF REGARDING OBJECTIONS TO GOOGLE'S PROPOSED CLAIM CONSTRUCTIONS** Dept.: Courtroom 8, 19th Floor Judge: Honorable William H. Alsup

INTRODUCTION

Google's brief makes clear that its proposed constructions aim to avoid infringement. But the intrinsic evidence controverts Google's narrowing proposals, which should be rejected.

ARGUMENT

I. OBTAIN A REPRESENTATION OF AT LEAST ONE CLASS FROM A SOURCE DEFINITION PROVIDED AS OBJECT-ORIENTED PROGRAM CODE ('720 PATENT)

Claim	Term or Phrase	Oracle Proposed Construction	Google Proposed Construction
'720 patent, Claims 1 and 10	obtain[ing] a representation of at least one class from a source definition provided as object-oriented program code	No construction necessary. The phrase has the ordinary meaning that its constituent words give it.	load at least one class definition by compiling object oriented source code

Google's brief perpetuates its error in equating "source" in the "obtaining" step of Claims 1 and 10 with the technical concept of "source code." The '720 patent uses the word "source" to mean "a point of origin or procurement," which is its plain and ordinary meaning. MICROSOFT PRESS COMPUTER DICTIONARY 443 (3d ed. 1997) (definition of "source": "In information processing, a disk, file, document, or other collection of information from which data is taken or moved.") (Declaration of Marc Peters in Support of Oracle's Objections to Google's Proposed Claim Constructions (ECF No. 646) ("Peters Decl.") Ex. A.)

Google's proposed construction is directly contradicted by the '720 specification. Claims 5 and 14 require that the "source definition" be maintained as a class file. But even Google acknowledges, as it must, that a "class file" is a compiled ***binary*** file rather than source code. (ECF No. 647 at 3.) Claims 5 and 14 are original claims and so part of the specification, forming the written description of the invention. They are not "anomalies" to be cast aside, as Google does without any legal authority. (*Id.* at 6.)

Google's argument rests on specification language and dependent claims that refer, not to the "obtaining" step it seeks to construe, but to a different claim limitation: the "interpreting and instantiating" step, which is not at issue here. The first three steps of Claim 10 are reproduced here in full to illustrate this point (Claim 1 has analogous structural limitations):

> 10. A method for dynamic preloading of classes through memory space cloning of a master runtime system process, comprising:
>
> executing a master runtime system process;
>
> obtaining a representation of at least one class from a source definition provided as object-oriented program code;
>
> interpreting and instantiating the representation as a class definition in a memory space of the master runtime system process; . . . .

Google's reliance on a ***single*** reference to compilation in the body of the specification—"an instance of the class is created by compiling the source and the class instance is installed in the system class dictionary (block 160)"—is thus misplaced. (ECF No. 647 at 4.) This sentence is part of the written description of the "interpreting and instantiating" step and not of the "obtaining" step. This is confirmed by Figure 10 of the '720 patent, in which step 157 is the "obtaining" step ("load bytes for class") and step 160 is the "interpreting and instantiating" step ("create instance of class"). There is no description of the "obtaining" step in the specification that includes compiling source code.

Likewise, the dependent claims that Google cites in its brief do not limit the "obtaining" step to compiling source code. Google is wrong when it states (ECF No. 647 at 6) that the "class definition" in Claims 2 and 3 is a class file, in order to argue that a "source definition" may not be a class file. The complete phrase in Claims 2 and 3 is "***instantiated*** class definition," which refers to an already-interpreted and instantiated "class object to represent the class in memory" according to the specification. '720, 6:54. This is the output of the "master runtime system process to interpret and to instantiate the representation as a class definition in a memory space of the master runtime system process" in Claim 1, which provides the antecedent basis for the phrase "the instantiated class definition" in Claims 2 and 3. The phrase "class definition" does not appear in the "obtaining" phrase that Google seeks to construe.

Nor does Claim 8 support Google's proposed construction. Claim 8 requires that the "object-oriented program code is written in the Java programming language." Of course, Java class files were originally written in the Java programming language and were created by a Java compiler, as the specification of U.S. Patent No. 7,213,240, which is incorporated by reference

into the '720 specification, makes clear. (The '240 specification refers to "class files" maintained on a server as being "programs written in a platform-independent language, such as Java," as explained on pages 4-5 of Oracle's Objections to Google's Proposed Claim Constructions ("Oracle's Opening Brief") (ECF No. 645).) Thus, the specification confirms that the claimed "obtaining" step includes obtaining an already-compiled class file, not source code. '240, 9:46-60 (Peters Decl. Ex. C). Google's proposed interpretation of unasserted Claim 8—that the reference to Java means "source definition" equals "source code"— is contrary to this disclosure. (ECF No. 647 at 6.)

Google's argument that the plain meaning of the "obtaining" phrase fails to give meaning to all of the claim language and renders portions of it superfluous is incorrect. (ECF No. 647 at 6.) Whether Claim 1's recitation of "class preloader" standing alone covers a traditional Java class loader configured to preload classes from class files is irrelevant. The inclusion of the phrase "to obtain a representation of at least one class from a source definition provided as object-oriented program code" aids in understanding the claim, and that is a sufficient purpose. *Bell & Howell Document Mgmt. Prods. Co. v. Altek Sys.*, 132 F.3d 701, 707 (Fed. Cir. 1997) ("[D]efining a state of affairs with multiple terms should help, rather than hinder, understanding. Being 'integrally bonded' and 'free of adhesive' are mutually reinforcing definitions rather than being superfluous."); *Infosint, S.A. v. H. Lundbeck A/S*, 603 F. Supp. 2d 748, 757 (S.D.N.Y. 2009) (when specification and claims make construction of word clear, use of additional words "to help not hinder understanding" is not superfluous). The words in the "to obtain…" phrase in Claim 1 also serve the purpose of providing an antecedent basis for further claim limitations in dependent Claims 5 and 8; no dependent claim otherwise further limits "class preloader."

Google's two cases on "superfluous" claim language (*Merck* and *Haemonetics*) are distinguishable. The Federal Circuit rejected the district court's constructions because they were directly contrary to the claim language and were not the ordinary and customary meaning as used in the specification. *Haemonetics Corp. v. Baxter Healthcare Corp.*, 607 F.3d 776, 781 (Fed. Cir. 2010) (when claim recited "centrifugal unit comprising a centrifugal component and a plurality of tubes" and specification disclosed corresponding embodiment, "centrifugal unit" could not be

construed to lack tubes); *Merck & Co. v Teva Pharms. USA, Inc.*, 395 F.3d 1364, 1372 (Fed. Cir. 2005) ("about" had its ordinary meaning of "approximately," and not "exactly" as construed by district court, because patentee had not acted as lexicographer). In cases where the proper construction is the ordinary definition, such as in *Bell & Howell* and *Infosint*, additional claim terms with reinforcing meanings are "helpful" and not superfluous. It is also significant that the phrase "class preloader" does not appear in Claim 10, and so Google's "superfluous" argument cannot apply to it.

Google's proposed construction is based only on extrinsic materials beyond the specification. The phrase "source code" appears nowhere in the '720 patent. It was known in the art that bytecode (contained in class files) could be compiled. *See* '205, 2:1-13 (filed before the '720). When the '720 refers to compiling, it just as much refers to compiling bytecode as it does to compiling source code. The specification states that the exemplary routine for preloading classes will first "load the bytes for the class from the source associated with the applicable bootstrap class loader 39 and system application class loader 40" ('720, 9:49-51) and only ***afterwards*** "an instance of the class is created by compiling the source" ('720, 9:52-54). The use of the word "bytes" means that the representation of the class is in the form of bytecodes (class files), not source code. *See* Java Virtual Machine Specification Ch. 4 (1996) ("This chapter describes the Java Virtual Machine class file format. . . . A class file consists of a stream of 8-bit bytes."), *available at* http://java.sun.com/docs/books/jvms/first_edition/html/ClassFile.doc.html. Google has provided no evidence that persons of ordinary skill in the art ever used the word "bytes" as shorthand for "source code."

It bears repeating that Google's argument to the Court is directly contrary to the position it took in the '720 *inter partes* reexamination. In the PTO, Google argued that class files were the claimed "source definition provided as object-oriented program code." (Peters Decl. Ex. D at 22 & Ex. 17.) But here, Google takes the opposite position and argues that a class file cannot possibly be the "source definition provided as object-oriented program code" from which a class preloader obtains a representation of a class when the class is to be loaded. Google has yet to inform the PTO that it changed its position.

There is no need to construe the phrase "obtain a representation of at least one class from a source definition provided as object-oriented program code," which has the ordinary meaning of its constituent words. Obtaining a class representation from a source definition includes "identifying a ***binary form*** of a class type as identified by specific name." '720, 6:49-50 (emphasis added). Rejecting Google's attempt to narrow this term does not require the Court to rephrase it for the jury. At most, the Court should instruct the jury that a "source definition" may be in binary or source code form. The ordinary meaning of the term embraces both.

II. RUNTIME ('205 PATENT)

Claim	Term or Phrase	Oracle Proposed Construction	Google Proposed Construction
'205 patent, Claim 1	runtime	No construction necessary. The ordinary meaning is "during execution of the virtual machine." (per 2/22/2011 Joint Claim Construction Statement)	during execution of the virtual machine instructions (per 2/22/2011 Joint Claim Construction Statement)

Google attempts to redefine "runtime" by taking only a narrow slice of runtime—execution of "the" virtual machine instructions—and excluding everything else that occurs at runtime. This is not supported by the specification or the claim language.

Nothing in Claim 1 of the '205 patent limits "runtime" to the execution of any particular virtual machine instructions. The preamble of Claim 1, which describes the context for the invention but does not limit it, states that the invention is "a method for increasing the execution speed of virtual machine instructions at runtime." Oracle agrees that when virtual machine instructions are executed, it is at runtime. But the inverse is not true: many things may happen at "runtime" other than the execution of particular virtual machine instructions. Indeed, Claim 1 claims an invention when the "generating, at runtime" and "executing" steps are ***separate*** steps:

> 1. In a computer system, a method for increasing the execution speed of virtual machine instructions at runtime, the method comprising:
>
> receiving a first virtual machine instruction;
>
> generating, at runtime, a new virtual machine instruction that represents or references one or more native instructions that can be executed instead of said first virtual machine instruction; and

> executing said new virtual machine instruction instead of said first virtual machine instruction.

See Phillips v. AWH Corp., 415 F.3d 1303, 1314 (Fed. Cir. 2005) (*en banc*) ("To begin with, the context in which a term is used in the asserted claim can be highly instructive."). All that Claim 1 requires is that the system generate a new virtual machine instruction ***before*** it executes the new instruction. The claim does not require that the generation be performed during the execution of "the" virtual machine instructions, or any virtual machine instruction or instructions, for that matter. The generation of new instructions could be performed when a class is loaded into a virtual machine (which happens at runtime) and before any instructions of the class are executed. The only execution that the claim requires follows the "generating, at runtime" step.

The specification of the '205 patent does not limit "runtime" to the execution of any particular virtual machine instructions. Code execution does occur at runtime, but the '205 patent makes clear that ***native*** machine instructions may be executed:

> In general, embodiments of the present invention provide innovative systems and methods for increasing the execution speed of computer programs executed by an interpreter. A portion of a function is compiled into at least one native machine instruction so that the function includes both virtual and ***native*** machine instructions ***during execution***.

'205, 2:35-40 (emphasis added). The compiling performed by this embodiment need not occur during execution of the virtual machine instructions; it may occur before they are executed, so that when execution does occur, both virtual and native instructions are executed. This is the best understanding of the embodiment, because when the compiler is executing, the virtual machine instructions are not—and yet it is still "runtime" under the ordinary meaning of the word. *See also* '205, Abstract ("A portion of the virtual machine instructions of the function are compiled into native machine instructions so that the function includes both virtual and native machine instructions. ***Execution of the native machine instructions*** may be accomplished by overwriting a virtual machine instruction of the function with a virtual machine instruction that specifies execution of the native machine instructions.") (emphasis added).

Other examples of activities performed at runtime disclosed in the specification that are not execution of "the" virtual machine instructions include initialization, class loading,

verification, compiling, linking, bytecode table generation, and snippet generation, as explained in Oracle's Opening Brief (ECF No. 645 at 10-13). When a program implementing the inventive method of Claim 1 performs the "generating" step, it is at runtime, but "the" virtual machine instructions are not being executed. Instead, the instructions of the program are being executed. Google's argument thus fails because it is contradicted by the specification.

Google's proposed construction is ambiguous because it leaves open the question: which are "the" virtual machine instructions? They are not in the body of the claim, which recites only "a first virtual machine instruction" which is never executed ("executing said new virtual machine instruction ***instead of*** said first virtual machine instruction"). The Court should reject a construction that invites additional questions and risks having to "construe the construction." (So the Court is aware, the reason for the ambiguity is that Google wants to avoid its admissions in the Android documentation that its accused "dexopt" program requires information "only available at runtime" and argue that "runtime" doesn't mean "runtime.")

Google's citations to the specification in its brief stand only for the noncontroversial proposition that when virtual machine instructions are being executed, it is runtime. (ECF No. 647 at 8 (citing '205, 1:17-21 ("porting an existing computer program to run on a different computer platform"); 1:47-51 ("An advantage of utilizing virtual machine instructions is the flexibility that is achieved since the virtual machine instructions may be run, unmodified, on any computer system that has a virtual machine implementation, making for a truly portable language."); 9:55-57 ("the faster the program will run")).) But Google's citations do not stand for the proposition that the '205 patent has redefined "runtime" to ***only*** occur when some particular yet unspecified virtual machine instructions are being executed. The patentee did not act as a lexicographer to give the general term "runtime" such a limited meaning, and the ordinary meaning controls.

CONCLUSION

For the foregoing reasons, Oracle respectfully requests that the Court decline to construe the two terms in dispute.

Dated: January 11, 2011

MICHAEL A. JACOBS
MARC DAVID PETERS
DANIEL P. MUINO
MORRISON & FOERSTER LLP

By: /s/ Michael A. Jacobs

Attorneys for Plaintiff
ORACLE AMERICA, INC.